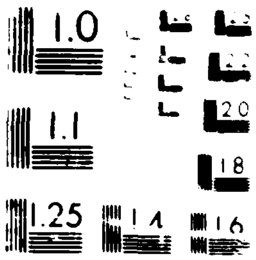
AD-A183 575 DESCRIPTION OF A LIMITED VAX-ADA PROGRAM SUPPORT ENVIRONMENT DEVELOPED WI..(U) ADMIRALTY RESEARCH ESTABLISHMENT PORTLAND (ENGLAND) M I HARVEY MAY 87 1/1

UNCLASSIFIED ARE-TM-(UDP)-87103 DRIC-BR-102676 F/G 12/5 NL

END
DATE
FILMED
9 87


1.0
1.1
1.25
1.4
1.6
1.8
2.0

UNCLASSIFIED/UNLIMITED

2

ARE TM (UDP) 87103

AD-A183 575

DTIC

ARE TM (UDP) 87103

ACCN No 75449

MAY 1987 34

# DESCRIPTION OF A LIMITED VAX-ADA PROGRAM SUPPORT ENVIRONMENT DEVELOPED WITHIN UDP DIVISION VERSION 1.0

by

MT HARVEY

ACCN No 75449

ADMIRALTY RESEARCH ESTABLISHMENT
Procurement Executive, Ministry of Defence,
Southwell, Portland, Dorset.

UNCLASSIFIED/UNLIMITED

87 8 18 079

## AMENDMENTS

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| NUMBER | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| DATE | | | | | | | | | | |

ARE TM (UDP) 87103
MAY 1987

# Description of a Limited VAX-Ada Program Support Environment Developed within UDP Division
Version 1.0

M.T. Harvey

UDP Division
A.R.E.
Portland
Dorset

## Abstract

This report describes a limited program support environment, designed within ARE, for use with the DEC VAX-Ada compiler. It describes the environment structure, how it should be used, and gives details of all the available tools.

## Modification Record

VERSION 1.0

This is the first release of this software package and its associated documentation.

## CONTENTS

## 1 INTRODUCTION

From the early design stages of the Ada programming language it has been clear that it could be used in the same way as a conventional programming language. However, it is widely believed that Ada, in common with many other languages, could be greatly improved by being incorporated into a support environment which would aid the development, maintenance and management of a software project written in Ada.

The DEC VAX-Ada product along with the VMS operating system does provide many of the tools required to form a program support environment, but many of these tools are not specifically designed for this purpose and lack many desirable configuration control and development facilities.

The UDP Ada environment is a limited program support environment setup for use with the VAX-Ada compiler, it provides additional features which are not directly available from DEC's VAX-Ada compiler or the ACS (Ada Control System) utility. The environment has its own directory structure (database) and commands which should be used when developing Ada programs on the VAX.

Ideally the tools used in a support environment for the Ada language should be written in Ada and contain a high degree of portability. However, since the DEC VAX-Ada compiler is only targeted for VAX machines the portability of the surrounding environment is not so critical. At present all the UDP command procedures are written in DCL (Digital Command Language). The environment is therefore suitable for use on any VAX machine using the VMS operating system.

This document describes how the UDP program support environment works, how it should be used and gives details of all the tools (command procedures) associated with it. Most of the information contained in this report can also be obtained by the use of the online help facility - UDP_HELP.

## 2 DATABASE STRUCTURE

For a detailed example of the structure formed when a new database is created refer to figure 2.2. The block diagram in figure 2.1 gives an indication of the general format of the database.

Figure 2.1

Block Structure of Database

```
                    ---------                         ENVIRONMENT
                   | ENV_    |-------------------------- LEVEL
                 / | NAME    | \
                /   ---------   \
               /        |        \
              /         |         \
     ---------       ---------     ---------              WORK AREA
    | USER_1  |     | USER_2  |   | MASTER    |------------ LEVEL
    |         |     |         |   |           |
     ---------       ---------     ---------
    / | | | \       / | | | \     / | | | \
   /                                       \
 ---------                              ---------          WORKING
| SOURCE  |                            | ADALIB  |----- LEVEL
|         |                            |         |
 ---------                              ---------
/    |    \
```

### 2.1 Explanation Of Database Levels

The three environment levels shown in figure 2.1 are described in this section.

#### 2.1.1 Environment Level -

This level contains a single directory which is given the name of the environment, it contains the directories specified in the work area level and the environment login command file, it should not contain any other files. If the project has more than one user then this level must be a top level directory.

#### 2.1.2 Work Area Level -

This level contains the directories which represent the different work areas in the project. Every project must have a MASTER work area which contains only working software and fixed documentation. The master directory contains the top level Ada library, all other work area's contain Ada sub-libraries (see Developing Ada programs manual for details of sub-libraries). If the project has more than one user then each work area is owned by a particular user. Access for other users will be limited to read and execute. The master work area should be owned by the project manager. It is perfectly acceptable for one user to have more than one work area. Sub-directories at this level contain the work area login command file and the sub-directories specified in the working level. To move to this level directory type TOP.

### 2.1.3 Working Level -

This level contains all the files created by the project. The initial directories setup for the user are shown below:

(a) ADALIB - This directory contains either a VAX-Ada top level library (master work area) or a VAX-Ada sub-library (normal work area).

To move to this directory type ADAL*IB.

(b) SOURCE - This directory should contain all the Ada source code used within the work area. It is possible that the user may wish to create further sub-directories for test software, different versions, package specifications etc.

To move to this directory type SOUR*CE.

(c) LIST - This directory contains all the list files generated by the Ada compiler and all the map files created by the VAX linker.

To move to this directory type LIST.

(d) EXECUTE - This directory contains the executable Ada code generated by the VAX linker, it should also contain any dump files created as a result of run time errors.

To move to this directory type EXEC*UTE.

(e) COMMAND - This directory contains all the users command files, along with a file called LOGICAL.COM which is used by some of the UDP commands.

To move to this directory type COMM*AND.

(f) SCRATCH - This directory contains any temporary work files which may be created by either UDP commands or by the users own command files.

To move to this directory type SCRA*TCH.

(g) REPORTS - This directory contains all the documentation relevant to the particular work area. The user may wish to create further directories at this level eg. one for system documentation and one for software documentation.

To move to this directory type REPO*RTS.

(h) TEST_DATA - This directory contains all test data relevant to the particular work area.

To move to this directory type TEST*_DATA.

(i) BATCH_LOGS - This directory contains all the log files generated from batch jobs submitted from this work area.

To move to this directory type BATL*OGS.

## 2.2 Example Of Database Structure

The initial structure of a typical database used in a UDP Ada envirionment is shown in figure 2.2. The database should be created by the system manager using the UDP_ENV_CREATE command. This database has the folowing features:

1. The project is called THE_PROJ.

2. There are two users currently working on the project (HARVEY and MSUACC).

3. Both users have the THE_PROJ identifier which associates them with the project.

4. HARVEY has been designated as the project manager.

5. The protection of directories and files in the database are such that:

   (a) Both users have read and execute (RE) access to the entire database; they have the THE_PROJ identifier (THE_PROJ RE).

   (b) MSUACC has read, write, execute and delete (RWED) access to the [.MSUACC] work area (MSUACC RWED).

   (c) MSUACC has only RE access to the [.MASTER] and [.HARVEY] work areas (THE_PROJ RE).

   (d) HARVEY has RWED access to [.HARVEY] work area and, as project manager, RWED access to the [.MASTER] work area (HARVEY RWED).

   (e) HARVEY has only RE access to the [.MSUACC] work area (THE_PROJ RE).

   (f) No other users have any access to the database ([*,*] none).

**6. After logging in to their own account on the VAX, the users may position themselves into any work area by executing the environment login command file and typing the appropriate work area name (typing the work area name will run the work area login command file). Examples of typical login procedures for the two users are shown in figure 2.2 (note it will be rare for MSUACC to enter any other work area than his own because he will only have RE access to them).**

**7. If a user wishes to use any of the UDP compilation or link command then the environment login command file and the appropriate work area login command file must be run from within the users LOGIN.COM file.**

**8. On creation of the initial database a single work file called ADALIB.ALB is created into each VAX-Ada library and sub-library. This file contains information about the contents of the library.**

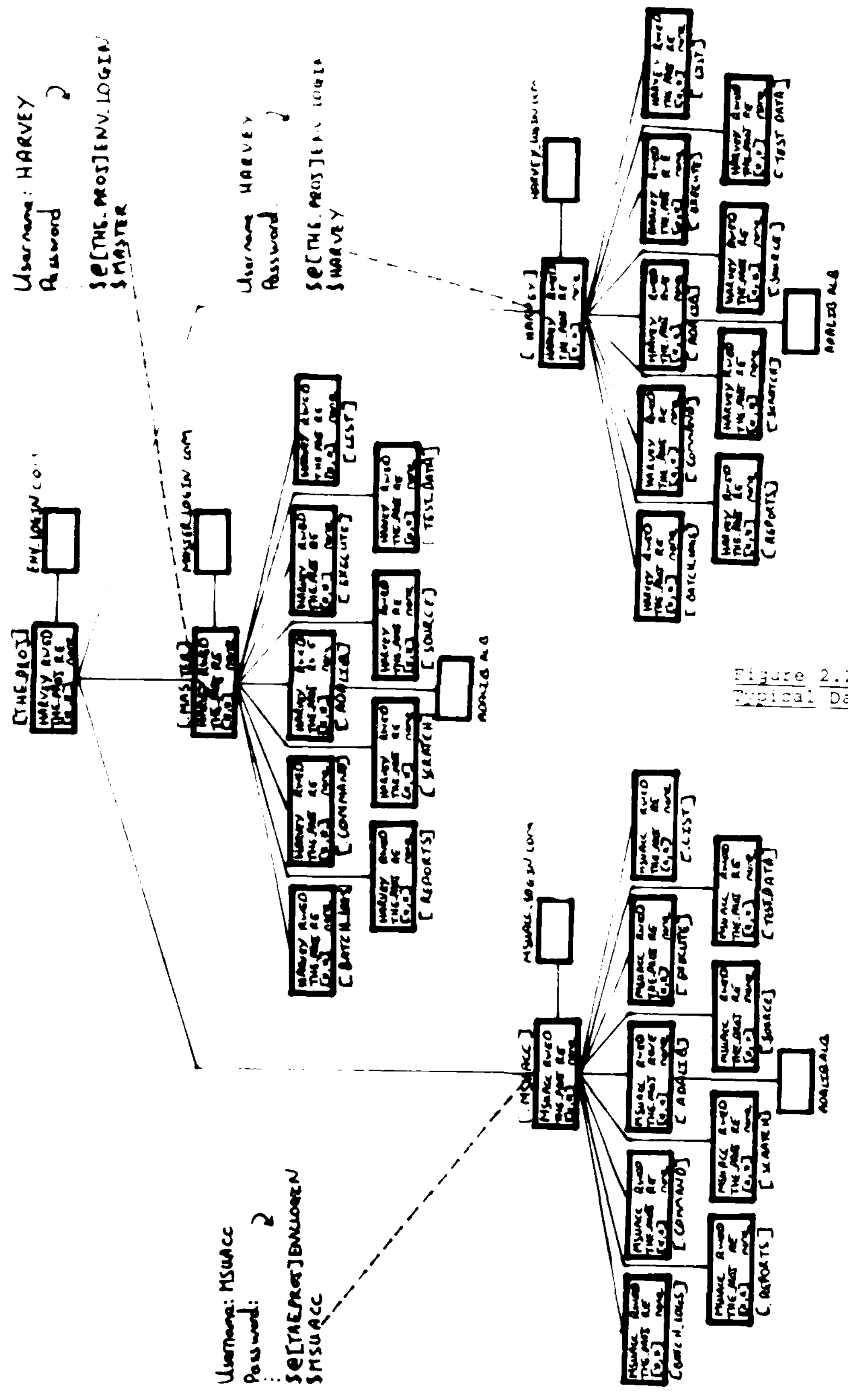


Figure 2.2
Typical Database

## 3 DEVELOPMENT STRATEGY

### 3.1 Introduction

This environment is intended to provide support for the development, maintenance and management of a software project throughout its lifecycle, in accordance with the aims of the Ada programming support environment (APSE).

The aspects of this environment which particularily support the above aims are:

(a) The database has a hierarchical structure using VMS directories.

(b) It has designated directories for each type of file used in the project.

(c) There is access control on all files and directories contained in the database (only multi-user project).

(d) There are individual programmer work areas and a master work area for all tested software and documentation.

(e) Configuration control features associated with development commands.

## 3.2 Getting Started

1. For a multi-user project the project manager must ask the VAX system manager to create an initial database, to his specifications, using the UDP_ENV_CREATE command. If the project has only one user then he may use the UDP_ENV_CREATE himself.

2. To work in a particular project a user must run the projects ENV_LOGIN.COM command file. Normally this file should be executed from within the user's LOGIN.COM file, this is because the users LOGIN.COM file is the only users command file executed when a batch job is submitted. Most of the UDP commands use batch jobs.

3. To work in a particular work area a user must run the appropriate work_area_LOGIN.COM file. This can be done by typing the name of the work area. Again this command file should be normally be executed from within the users LOGIN.COM file.

4. To create a file in a particular directory of a work area, the user must move to that directory typing the appropriate command word (eg. SOURCE to move to source code directory), more details of these command words and their abbreviated forms are given in section 2.1.3. The user can now create and edit files in the conventional way using the editor.

5. Once an Ada source file has been created the user may use one of the UDP Ada compilation commands to compile the Ada compilation unit(s) using the Ada batch queue. The output files from a compilation can be examined by moving the appropriate directory and any corrections can then be made.

### 3.2.1 Getting Started Example -

This example on how to get started uses the initial database shown in figure 2.1. The initial database has been setup, using the UDP_ENV_CREATE command, by the system manager to the project managers specifications. The project manager (HARVEY) has informed the user (MSUACC) that he must create and compile an Ada package specification called BASED_IO in the work area allocated to him.

The procedure which MSUACC should follow is:

1. Login to his own account:

```
Username: MSUACC
Password:
```

2. Edit his LOGIN.COM file and include command lines which will execute the environment login command file and his work area login command file; execute login command file:

```
$ EDIT LOGIN.COM

   ---------------------------------------------
   | .                                         |
   | .                                         |
   | .                                         |
   | $! Execute environment login command file |
   | $    @[THE_PROJ]ENV_LOGIN                 |
   | $!                                        |
   | $! Execute work area login command file.  |
   | $    MSUACC                               |
   | .                                         |
   ---------------------------------------------

$ @LOGIN
```

Note that the above edit and the 'manual' execution of the login command file only needs to be carried out the first time MSUACC uses this project.

3. Move to source directory and create Ada source file called BASED_IO_SPEC.ADA:

```
$ SOURCE
$ EDIT BASED_IO_SPEC.ADA

   ---------------------------------------------
   | package BASED_IO is                       |
   | .                                         |
   | .                                         |
   | end BASED_IO;                             |
   ---------------------------------------------
```

4. Compile BASED_IO_SPEC.ADA and wait for mail message:

```
$ UDP_ADA BASED_IO_SPEC
```

5. If the mail message says that the compilation was successful then the user's Ada program sub-library will be updated to include the BASED_IO *package specification*. After suitable testing of the the unit, the user (MSUACC) may notify the project manager (HARVEY) that the package specification BASED_IO is completed and that it can be placed into the master work area (this notification should be made using the UDP_NOTIFY command).

6. If the mail message says that the job has failed or no mail message is received then the user should move to the BATCH_LOGS directory and type the appropriate log file to examine what commands were executed and what system errors may have occured.

```
$ BATLOGS
$ TYP BASED_IO_SPEC.LOG
  .
  .
  .
```

7. If the batch log file shows that a compilation error has occured then the user should move to the list directory and examine the appropriate list file.

```
$ LIST
$ TYP BASED_IO_SPEC.LIS
  .
  .
  .
```

8. The user should now correct any errors and resubmit the compilation if neccessary and repeat the above process (from 3 to 7) as required.

## 3.3 Environment Creation

Before the UDP_ENV_CREATE command is used to create an initial database the designated project manager must decide upon a name for the project, this name will be used for the top level directory for the database.

The project manager must also consider how many work area's are required in the database. In many cases it is likely that there will be single work area for each person working on the project. However, the allocation of work areas should depend on the distribution of work in the project and it is therefore acceptable for a single user to have two or more work areas. If the project using the Ada compiler is to be a multi-user project then the project manager must ask the VAX system manager to setup the database under a project account. If, however, the project has only a single user then he can set up database himself using the environment create command (UDP_ENV_CREATE).

## 3.4 Software Development

All software generated in the database should be developed and tested in the user's work area before being incorporated into the master work area. Once in the master work area the software can be built into the overall system. The design of the DEC Ada library system is such that any software compiled into a user's sub-library it cannot be seen by any other user. Once software has been compiled into the master directory it can be seen and used by any user who's work area contains an Ada sub-library (the parent library is always program library in the master work area).

It is strongly recommended that all the source files for any software developed in the user's work area are copied into the master work area and that the software is then compiled into the master library. This is preferred to using the ACS MERGE command because it ensures that all the source code is contained in the master work area and that the master library is in a known state. This configuration control helps the maintenance of the project. If it is found that software contained in the master directory has to be altered then it should first be copied back into a users work area. A compiled unit in a sub-library will be used in preference to a unit of the same name in the master library. Changes to files must not be made while they are contained in the master work area.

Once a working system has been produced and the system is to be released a copy of the entire master work area should be made to form a "frozen" version of the project. Further alterations to the system must then be incorporated into a later release. Please note that at present there is not a tool for automatically creating a frozen version of the master work area, but this is planned for a later release.

## 3.5 Extensions To Database

On creation of a new environment the user is provided with a number of directories, an environment login command file and work area login command file for each work area. Together these command files provide all the assignments neccessary for the user's to move about their database and develop software in the appropriate work areas. Additional work area's can be produced at any time as the need arises.

There is also a possibility that a user may wish to extend an existing work area by creating a number of further directories, provision is made for this but these additional directories should be limited in number, to keep each work area down to a controlable size. If a single work area is becoming too complex then the creation of further work areas, rather than additional directories, should be considered. The environment and login command files may also be altered to include user's commands but any changes made should be fully documented.

## 3.6 Use Of UDP Commands

To make full use of the facilities available within this environment it is very important that whilst the user may use most of the ACS commands available, he should not use the DCL or ACS commands which have been replaced by UDP commands. For examples the use of the DCL Ada command rather than its UDP_ADA equivalent will lead to the following problems:

1. Compilations can be carried out on-line rather than from the Ada batch queue. This means that swapping is likely to occur because the users working set sizes are not setup to cope with Ada compilations, swapping will increase the time of the compilation and greatly decrease the efficiency of the machine for other users.

2. The UDP_ADA command calls a command file which ensures that:

   (a) The source code is taken from the directory specified by the UDP_SET_SOURCE command.

   (b) the current Ada program library is used.

   (c) Output files generated are directed to specific directories.

```
ie.     Object code    - .ADALIB
        List file      - .LIST
        Batch log file - .BATLOG
```

Similar problems are caused if the ACS COMPILE, ACS RECOMPILE or ACS LINK commands are used.

## 4 FILE PROTECTION AND MANAGEMENT

Within a multi-user environment (ie. project account) the file protections are setup so that the owner of a particular work area has, by default, read, write, execute and delete access to all the files within it. The designated project manager owns the master work area and is the only user with read, write, execute and delete access to it. All users with the environment identifier have, by default, read and execute access to all areas of the environment which he does not own. In practice, because of a peculiarity of the DEC Ada library system, all users must have write access to the program library in the master work area. It is, however, essential that only the project manager writes to the master program library.

In a single user environment the user should use his environment in the same way as he would a multi-user environment. The file protections cannot be setup to prohibit access to files in different work areas in the database but the user should discipline himself to behave as if access was controlled. The user must act as both general user and project manager, but not at the same time.

In future releases of this system it is hoped that the access to files and directories within the environment can be made more rigourous, thereby providing greater configuration control .

## 5 PORTABILITY OF ENVIRONMENT

The Ada programming language was designed with great importance attached to the portability of the language. However, the DEC VAX-Ada compiler was designed specifically for use on the VAX family of machines. The VAX-Ada system exploits many aspects of the VAX architecture. All the tools, in the form of command files, used by the UDP program support environment are written in DCL (Digital Command Language). The alternative of writing all the tools in Ada was investigated, and is still a possibility for future releases, but the time available, the complexity of the work involved and the fact that the DEC VAX-Ada compiler can only be used on VAX machines, led to the decision to use DCL for developing the tools.

In general the tools available could be used on any VAX running VMS without any modifications. However, in this release, the directory structure and names must remain the same, and the following files must be ammended so that the correct disk is used:

```
UDP_ENV_CREATE.COM
UDP_ADA_LOGIN.COM
ENV_LOGIN.COM

eg. substitute DRB0: for DUA0:
```

At present the UDP Ada program is only suitable for use with the DEC VAX-Ada compiler, however with relatively small modifications it could also be used wtih other Ada compilers and for mixed language development.

## 6 FEATURES OF FUTURE RELEASES

The possibility of introducing some or all of the following features into future releases of the UDP Ada program support environment are being considered:

1. Correction of any errors in the current release.

2. Improvements to database directory structure.

3. Increased number and variety of tools, leading to less reliance on pure DCL commandS.

4. Greater configuration control.

5. Some tools written purely in Ada.

6. Formalised software release package and installation procedure.

7. Complete portability between VAX machines running the VMS operating system.

## 7 ENVIRONMENT COMMAND FILES

When an environment is created it contains two different types of command file. The environment login command file (ENV_LOGIN.COM) contains all the general commands needed to use the environment. The work area login command files (work_area_name_LOGIN.COM) contain all commands which are specific to a particular work area.

### 7.1 Environment Login Command File

This command file runs the UDP Ada login command file (which sets up all the UDP Ada commands), and allocates logical names for the environment disk and each user's work area login command file. This command file must be run from the users top level login command file if he wishes to use any of the UDP commands which use the batch queue.

Note - The following command in your LOGIN.COM file can be used so that ENV_LOGIN.COM is only run in batch mode -

```
$ if f$mode() .eqs. "BATCH" then @[env_name]ENV_LOGIN.COM
```

#### 7.1.1 Example Of Environment Login Command File -

This is a typical example of the commands contained within the environment login command file.

```
$ @DRB0:[ADAFORMAT.ADA_COMMAND_FILES]UDP_ADA_LOGIN

$ ASSIGN DRB0: ADA_USER_DISK
$ SET DEF ADA_USER_DISK:

$ MASTER :== @ADA_USER_DISK:[ADAFORMAT.TEMP.MASTER]MASTER_LOGIN.COM
$ USER_1 :== @ADA_USER_DISK:[ADAFORMAT.TEMP.USER_1]USER_1_LOGIN.COM
```

### 7.2 Work Area Login Command File

This command file provides the user with simple assignments to allow ease of movement between sub-directories, sets up work area name symbol, source file location logical name and current Ada program library. It also places the user at the top of his work area.

The command file is given a name of the form work_area_name_LOGIN.COM. There is an assignment made in the environment login command file which allows the work area login command file to be executed by simply typing the name of the work area. See section 7.1.1 for example of typical assignments made.

This command file must be run from the users LOGIN.COM command file if he wishes to use any of the UDP commands which use the batch queue.

Note - The following command in your LOGIN.COM file can be used so that the work area login command file is only run in batch mode -

```
$ if f$mode() .eqs. "BATCH" then work_area_name
```

### 7.2.1 Example Of Work Area Login Command File -

This is a typical example of the commands contained within the work area login command file.

```
$TOP :==SET DEF ADA_USER_DISK:[ADAFORMAT.TEMP.USER_1]
$SOUR*CE :==SET DEF ADA_USER_DISK:[ADAFORMAT.TEMP.USER_1.SOURCE]
$LIST :==SET DEF ADA_USER_DISK:[ADAFORMAT.TEMP.USER_1.LIST]
$EXEC*UTE :==SET DEF ADA_USER_DISK:[ADAFORMAT.TEMP.USER_1.EXECUTE]
$COMM*AND :==SET DEF ADA_USER_DISK:[ADAFORMAT.TEMP.USER_1.COMMAND]
$SCRA*TCH :==SET DEF ADA_USER_DISK:[ADAFORMAT.TEMP.USER_1.SCRATCH]
$TEST*_DATA :==SET DEF ADA_USER_DISK:[ADAFORMAT.TEMP.USER_1.TEST_DATA]
$REPO*RTS :==SET DEF ADA_USER_DISK:[ADAFORMAT.TEMP.USER_1.REPORTS]
$ADAL*IB :==SET DEF ADA_USER_DISK:[ADAFORMAT.TEMP.USER_1.ADALIB]
$BATL*OGS :==SET DEF ADA_USER_DISK:[ADAFORMAT.TEMP.USER_1.BATCH_LOGS]

$TOP

$ACS SET LIBRARY ADA_USER_DISK:[ADAFORMAT.TEMP.USER_1.ADALIB]

$WORK_AREA_NAME :== ADAFORMAT.TEMP.USER_1

$if f$mode() .eqs. "BATCH" then goto BATCH_LABEL
$UDP_SET_SOURCE ADA_USER_DISK:[ADAFORMAT.TEMP.USER_1.SOURCE]

$BATCH_LABEL:
$@ADA_USER_DISK:['WORK_AREA_NAME'.COMMAND]LOGICAL.COM
```

## 8 UDP COMMANDS

Once the environment has been setup the user can use all the VAX-Ada ACS commands described in the DEC manual "Developing Ada programs on VAX/VMS", with the exception of the following commands:

| | |
|---|---|
| ADA | - Do not use this command |
| ACS COMPILE | - Do not use this command |
| ACS RECOMPILE | - Do not use this command |
| ACS LINK | - Do not use this command |

These DEC commands have been replaced by the following UDP commands.

```
ADA   ---------------> UDP_ADA
ACS COMPILE ---------> UDP_COMPILE
ACS RECOMPILE -------> UDP_RECOMPILE
ACS LINK ------------> UDP_LINK
```

The UDP commands are very similar to the origonal commands but they additionally provide the following services -

(a) Submit all compilation and link jobs to the appropriate batch queues.

(b) Direct all output files to specific sub-directories.

(c) Take source files from pre-defined sub-directories or ada program libraries.

The other UDP commands are -

1. UDP_ENV_CREATE

2. UDP_SET_SOURCE

3. UDP_NOTIFY

4. UDP_HELP

## 8.1 UDP_HELP

This command gives help on all the available UDP commands and details of many aspects of work in the UDP environment. Assistance can be gained on the following subjects.

Command_files, Development_Strategy, Directory_structure, Environment_creation, Introduction, Moving Directories, UDP_commands, UDP_ADA, UDP_COMPILE, UDP_ENV_CREATE, UDP_LINK, UDP_NOTIFY, UDP_RECOMPILE, UDP_SET_SOURCE

## 8.2 UDP ADA

This command submits a job to the Ada batch queue which will compile one or more ada source files using the DEC VAX-Ada compiler.

The command has two parameters:

```
UDP_ADA filename,...  (options)
```

1. Filename and positional qualifiers.
2. Options (all command qualifiers for Ada command available).

### 8.2.1 Parameters -

1. Filename - Only filename may be given here. Directory specification is obtained from the result of the previous UDP_SET_SOURCE command. If the source directory is changed while a compilation is on the batch queue the new source directory will be used and an error is likely to occur. All positional qualifiers used with the DCL Ada command may also be used here (type HELP ADA for more details).

2. Options - This is an optional parameter. All qualifiers used with the DCL ADA command may be used here.

### 8.2.2 Examples -

Typical Examples (SPACES ARE CRITICAL):

```
1. UDP_ADA MY_FILE /MACHINE_CODE

2. UDP_ADA MY_SECOND_FILE

3. UDP_ADA MY_FIRST_FILE/CHECK,MY_SECOND_FILE /ERROR_LIMIT=30

4. UDP_ADA
      What is the name of the file to be compiled ?..... MY_FILE
      What options do you require ? (eg /MACHINE_CODE).. /SHOW
```

## 8.3 UDP COMPILE

This command uses the ACS COMPILE command with the /COMMAND qualifier to create a command file containing details of all compilations required to make the closure of a particular unit up to date. The list of compilations is displayed and the user asked if he wishes to compile these units. If required the command file is then submitted to the Ada batch queue.

The command has two parameters:

UDP_COMPILE unitname (options)

1. Unit name (Ada name) and positional qualifiers.
2. Options (all qualifiers for ACS COMPILE command available)

### 8.3.1 Parameters -

1. Unit name - Ada unit name must be given here. All positional qualifiers used with the ACS COMPILE command may also be used here (use HELP ACS for more details).

2. Options - This is an optional parameter. All qualifiers used with the ACS COMPILE command may be used here.

### 8.3.2 Command File -

The command file created by the ACS COMPILE command has the name unit_name_COMPILE.COM this file is only automatically deleted by a further UDP_COMPILE command being issued for this unit therefore please note that:

1. Only one UDP_COMPILE command can be issued at a time for one particular ada unit. Further commands should not be issued until the batch job is completed.

2. The unit_name_COMPILE.COM files contain details of the Ada compilations caused by the ACS COMPILE command. Please delete them when they are no longer required.

### 8.3.3 Examples -

Typical Examples (SPACES ARE CRITICAL):

1. 
```
UDP_COMPILE MY_UNIT_SPEC /NODATECHECK/CLOSURE
```

2. 
```
UDP_COMPILE MY_UNIT
```

3. 
```
UDP_COMPILE MY_UNIT/BODY,OUR_UNIT  /NODATECHECK/CLOSURE
```

4. 
```
UDP_COMPILE
What is the name of the file to be compiled ?.... MY_UNIT_BODY
What options do you require ? (eg /MACHINE_CODE). /SHOW
```

## 8.4 UDP_RECOMPILE

This command uses the ACS RECOMPILE command with the /COMMAND qualifier to create a command file containing details of all compilations required to make the closure of a particular unit up to date. The list of compilations is displays and the user asked if he wishes to compile these units. If required the command file is then submitted to the Ada batch queue.

The command has two parameters:

```
UDP_RECOMPILE unitname (options)
```

1. Unit name (Ada name) and positional qualifiers.
2. Options (all qualifiers for ACS RECOMPILE command available)

### 8.4.1 Parameters -

1. Unit name - Ada unit name must be given here. All positional qualifiers used with the ACS RECOMPILE command may also be used here (type HELP ACS RECOMPILE for more details).

2. Options - This is an optional parameter. All qualifiers used with the ACS LINK command may be used here.

### 8.4.2 Command File -

The command file created by the ACS RECOMPILE command has the name unit_name_RECOMPILE.COM this file is only automatically deleted by a further UDP_RECOMPILE command being issued for this unit therefore please note that:

1. Only one UDP_RECOMPILE command can be issued at a time for one particular Ada unit. Further commands should not be issued until the batch job is completed.

2. The unit_name_RECOMPILE.COM files contain details of the Ada compilations caused by the ACS RECOMPILE command. Please delete them when they are no longer required.

### 8.4.3 Examples -

Typical Examples (SPACES ARE CRITICAL):

```
1. UDP_RECOMPILE MY_UNIT_SPEC /NODATECHECK/CLOSURE

2. UDP_RECOMPILE MY_UNIT

3. UDP_RECOMPILE MY_UNIT/BODY,OUR_UNIT  /NODATECHECK/CLOSURE

4. UDP_RECOMPILE
   What is the name of the file to be compiled ?.... MY_UNIT_BODY
   What options do you require ? (eg /MACHINE_CODE). /SHOW
```

## 8.5 UDP_LINK

This command uses the ACS LINK command with the /COMMAND qualifier to create a command file containing details of the link required to form the executable code for a particular sub-program body. The list of Ada units to be linked is displayed and the command file is submitted to the Ada batch queue.

The command has three parameters:

```
UDP_LINK unitname (foreign code) (options)
```

1. Unit name (Ada main procedure name) and positional qualifiers.
2. Foreign code (non-Ada code).
3. Options (all qualifiers for ACS LINK command available)

### 8.5.1 Parameters -

1. Unit name - Ada main procedure name must be given here. All positional qualifiers used with the ACS link command may also be used here (type HELP ACS LINK for more details).

2. Foreign code - This is an optional parameter. All non-Ada units to be included at link time should be included here.

3. Options - This is an optional parameter. All qualifiers used with the ACS LINK command may be used here.

### 8.5.2 Command File -

The command file created by the ACS LINK command has the name unit_name_LINK.COM this file is only automatically deleted by a further UDP_LINK command being issued for this unit, therefore please note that:

1. Only one UDP_LINK command can be issued at a time for one particular Ada unit. Further commands should not be issued until the batch job is completed.

2. The unit_name_LINK.COM files contain details of the required link resulting from the ACS LINK command. Please delete them when they are no longer required.

### 8.5.3 Examples -

1. 
```
UDP_LINK MAIN SYS$LIBRARY:CORLIB/LIBRARY /BRIEF
```
2. 
```
UDP_LINK MAIN
```
3. 
```
UDP_LINK
What is the name of the MAIN UNIT to be linked.. MAIN
What options do you require ? (eg /BRIEF)....... /SHOW
```

## 8.6 UDP_NOTIFY

This command file allows a user to send a mail message to any other specified user. The message is formed by editing a work file using the EDT editor.
The command has one parameter:

```
UDP_NOTIFY user_name
```

### 8.6.1 Parameter -

1. Username - This name must be a valid user name for a VAX user.

### 8.6.2 Examples -

1. UDP_NOTIFY MANAGER
2. UDP_NOTIFY
   Who do you want to send a message to ? HARVEY

## 8.7 UDP_SET_SOURCE

This command file allows the user to explicitly specify the directory containing the Ada source files to be used in batch compilations. This is done by altering the command file LOGICAL.COM.
The command has one parameter:

```
UDP_SET_SOURCE directory_specification
```

### 8.7.1 Parameter -

1. Directory_specification - This must be a full VMS directory specification.

### 8.7.2 Examples -

1. UDP_SET_SOURCE DRB0:[ENVIRONMENT.MASTER.SOURCE.BODIES]

2. UDP_SET_SOURCE
   In what directory is your Ada source code held ?

   DRB0:[ENVIRONMENT.MASTER.SOURCE.SPECS]

## 8.8 UDP ENV CREATE

This command file will create a working environment for both single user and multi-user projects. This environment is specifically designed for use with the DEC-Ada compiler.

The user/system manager is asked about various aspects of the environment and it is then created to the users specification. The command file can also be used to create a new work area within an existing environment.

### 8.8.1 Actions Of Command File -

The command file does the following:

1. Asks whether user wants a new environment to just to add new work area's to an existing environment.
2. Reminds the user that only the system manager can create a top level directory (eg [000000.MY_ENV]) and that this is neccessary if there is to be more than one user.
3. Asks what disk environment is (to be) on.
4. Asks for the name (to be) given to the environment.
5. Asks for project manager username. (multi-user only)
6. Asks for project new users usernames (multi-user only)
7. Grants project identifier to new users. (multi-user only)
8. Allocates disk quota to project. (multi-user only)
9. Creates environment level directory and environment login command file (new environment only).
10. Sets up ACL's. (new environment and multi-user only)
11. Asks or the number of programmer work area's required.
12. Creates the MASTER work area (top program library). (new environment only)
    (a) Appends environment login command file with MASTER symbol.
    (b) Creates directories required in work area.
    (c) Sets up ACL's. (multi-user environment only)

(d) Creates top level DEC-Ada program library.

(e) Edits and copies MASTER_LOGIN command file.

13. Creates programmer(s) work areas (program sublibrary).

(a) Appends environment login command file with user symbol.

(b) Asks for name of work area.

(c) Creates directories required in work area.

(d) Sets up ACL´s if multi-user environment.

(e) Creates DEC-Ada program sublibrary.

(f) Edits and copies work_area_LOGIN command file.

The command files created/adapted by this command file are as follows:

(i) Environment login command file - this contains symbol names set up to enable the user to run the work area login command files, and runs the UDP_ADA login command file.

(ii) Work area login command files - this contains logical names to allow the user to move around his work area and sets up initial defaults for a particular user.

### 8.8.2 Example -

This example shows a typical single user environment creation.

```
$ UDP_ENV_CREATE
Do you want a new environment (E) or a new work area (W).....? : E

If your environment is to be  created using top level directory (i.e.
multi-user project), then this command file must be run by the system
manager.

Do you need to see the system manager (Y*ES/N*O/M*ANAGER)....? : N
What disk is environment on (EG. DRB0:)......................? : DRB0:
Full directory specification (except device) required
eg [ADAFORMAT.MY_ENV]
What is your environment directory ......? : [010030.MIXED_LANG]
How many programmer work areas do you require................? : 1
working...
working...
%LIBCRE,Library DRB0:[010030.MIXED_LANG.MASTER.ADALIB] created
What is programmer work area to be called...................? : WORK_1
working...
working...
%SUBLIBCRE, Sublibrary DRB0:[010030.MIXED_LANG.WORK_1.ADALIB] created
%SUBLIBCRE, Sublibrary DRB0:[010030.MIXED_LANG.WORK_1.ADALIB] created
What is programmer work area to be called...................? : WORK_2
working...
working...
%SUBLIBCRE, Sublibrary DRB0:[010030.MIXED_LANG.WORK_2.ADALIB] created
%SUBLIBCRE, Sublibrary DRB0:[010030.MIXED_LANG.WORK_2.ADALIB] created
New environment directory structure has been created
```